Techniques for debugging

Patrick Farrell

May 29, 2007

Goals for the talk

- For programmers: figure out why your code isn't working
- For non-programmers: be able to help programmers more

An entomological taxonomy: different kinds of bugs

- Different causes
- Different symptoms
- Different difficulty
- Different techniques to fix them

Segmentation faults

- Usually the easiest to fix
- Caused by touching memory that's not yours
- Usually a null() pointer, or argument that's not present
- Best fix: gdb will tell you exactly what line is causing it

Using gdb: the basics

- Start with: *gdb $PROGRAM*
- Place breakpoints: *break*, *watch*
- Begin execution: *run $ARGUMENTS*
- See what's going wrong

Fortran runtime errors

- The fortran runtime library often tells you something's gone wrong
- But you want to find out why!
- *break _gfortran_runtime_error*

FLAborts

- Sanity checks put in place by programmers
- List things that shouldn't ever happen
- Usually indicate problem upstream of abort
- *break __fldebug__flabort_pinpoint_*

Uninitialised variables

- A subroutine takes in inputs and gives outputs
- If the inputs aren't set, you usually get nonsense out
- Can be tricky to identify if initialisation is conditional
- Best fix: Valgrind is your friend

Valgrind

- Runs your program in a virtual machine
- Catches loads of difficult-to-find bugs
- Uninitialised variables
- Memory leaks
- Downside: really really slow
- Run: *valgrind $OPTIONS $PROGRAM*

When real numbers aren't enough: NaNs and Infs

- NaN: not a number
- Result of undefined arithmetic operation, *e.g.* $\sqrt{-1}, \frac{0}{0}$
- Inf: represents infinity
- Result of dividing by zero
- Best fix: trap on floating point exceptions
- Compile with *-ffpe-trap=invalid*

Memory overruns

- Objects in memory are usually laid side-by-side
- What if you overrun an array?
- You poke whatever happens to be next to it
- Usual symptom: random nonsense in your data
- Best fix: electric fence

Using electric fence

- *export LD_PRELOAD=/usr/lib/libefence.so*
- Run under gdb
- Converts memory overruns to segfaults!

Memory leaks

- Allocate memory but never deallocate it
- Gradually builds up to waste memory
- Best fix: valgrind

Stack smashes

- Like memory overruns, but on the stack
- Not malloc'ed $\Longrightarrow$ even worse
- Symptoms: random crashes, general lunacy
- Cause: memory overruns of stack-allocated data
- Cause: not passing correct arguments to subroutines
- Best fix: blood, sweat and tears :-(

Using strace

- Kernel does all the work on the system
- Strace records what your program asks for
- Run: *strace -o /tmp/strace.log $PROGRAM*